State-Sensitive Computational Modeling

Adriana Compagnoni*
Department of Computer Science
Stevens Institute of Technology, NJ

Paola Giannini†
Computer Science Institute, DiSIT
Università del Piemonte Orientale, Italy

Christopher Kelley
Department of Computer Science
Stevens Institute of Technology, NJ

ABSTRACT

BIOSCAPE^P defines a new modeling language for a state-dependent stochastic simulation of parallel processes in three dimensional space. The contributions of the language are as follows. Normally a modeling language describes an initial concentration of entities after which all changes are driven by a simulation of reactions. We instead design **conditional simulation commands** which depend on a **global state**. Our contribution is a new command in the form `when` R `run` $A_1 \cdots A_n$, which will cause $A_1 \cdots A_n$ entities to be added to the system when property R in the context of a global state is satisfied. Commands and reactions are further equipped with **user-defined update functions** to produce **side effects** on the global state. The global state for simulation is defined to include at least a simulation clock to enable time dependent computation.

BIOSCAPE^P matches the realistic nature of experimentation by defining uncertainty from two sources: stochastic movement generating reactions on proximity, and probabilistic choice where an entity has the ability to be involved in more than one reaction.

To capture the richer notion of state dependent conditional commands, which need to be evaluated periodically, we must define a system of **multi-level semantics** broken into two parts, **World Level Semantics** and **Individual Level Semantics**. Both levels take turns to evaluate their respective domains such that World Level Semantics evaluates the aforementioned conditional commands, while Individual Level Semantics simulates reactions, entity movement, and the update of the simulation clock and timed entities.

1. INTRODUCTION

Computational models have long been used to generate virtual versions of laboratory experiments where initial concentrations react over a period of time producing time-stamped simulation results. Although this mechanism captures a large class of experiments, it does not model the conditional events that often alter elaborate wet-lab experiments.

Our earlier work on BIOSCAPE [4] helped us understand the needs of virtual experimentation in a 3D environment. Experimentalists, however, rely on time and other observable conditions to tailor their experiments, and for safety reasons, drugs are often administered in appropriate doses over time. Because of the lack of formalisms and tools to design virtual experiments depending on a global state including the simulation clock, we define BIOSCAPE^P. BIOSCAPE^P extends the syntax and semantics of BIOSCAPE to enable clock dependent simulation, and more generally state dependent computational modeling. Fig. 1 shows an example of a state dependent experiment. A simulation command is a collection of conditional events (`when-run`). The predicate guarding the `run` command may refer to global variables. In this example the variable `nBac` corresponds to the number of bacteria in the system. At time 0, initial concentrations of bacteria (`Bac`) and antimicrobial agents (`AmA`) are added to the system. The remainder conditional events prescribe to repeat every 12 hours the initial dose of antibiotics for 24 hours, if there are still bacteria present in the system (`nBac` > 1). `u` and $\mathtt{u}_1$ are user-defined state update functions that modify global variables depending on the relevant observations in each model. For example, they could keep track of the number of doses of antibiotics delivered.

$\mathtt{when}\ (\mathtt{clock} = 0)\ \mathtt{run}^{\mathtt{u}}\ \mathtt{Bac}_1 \ldots \mathtt{Bac}_n\ \mathtt{AmA}_1 \ldots \mathtt{AmA}_k;$
$\mathtt{when}\ (\mathtt{clock}{=}12hs \wedge \mathtt{nBac}{>}1)\ \mathtt{run}^{\mathtt{u}_1}\ \mathtt{AmA}_1 \ldots \mathtt{AmA}_k;$
$\mathtt{when}\ (\mathtt{clock}{=}24hs \wedge \mathtt{nBac}{>}1)\ \mathtt{run}^{\mathtt{u}_1}\ \mathtt{AmA}_1 \ldots \mathtt{AmA}_k;$

Figure 1: Antibiotics every 12 hours while infection persists

2. SYNTAX OF MODELING LANGUAGE

The syntax of BIOSCAPE^P appears in Fig. 2. We first give an intuitive introduction of the new syntactic constructs at the cost of introducing some implicit forward references.

We assume a *global state* V of variables $\mathtt{x}, \mathtt{y}, \ldots$ including the simulation clock, giving the programmer access to experimental time. Among the novel aspects of BIOSCAPE^P is a richer collection of commands made possible in part by the new concepts of experimental time and global state in a language-based modeling platform.

Since a number of language constructors have the ability to declare state modifications, we define *state update* functions. However we do not allow the user to update the simulation clock or the counters for the population of species, such as `nBac`. As we will see in Section 3.2, only simulation steps can update `clock` (See rule PR.TIME in Fig. 11).

DEFINITION 2.1 (STATE UPDATE). *Let $\mathcal{U} \subset dom(V)$ be the set of global variables that can be updated. A* state update *is a partial function $\mathtt{u} : \mathcal{U} \to \mathbb{Z}$. The* application of `u` to V*, written* $\mathtt{u} \circ V$*, is defined by:*

*Partially funded by the National Library Of Medicine of the National Institutes of Health under Award Number R01LM011826.
†Partly funded by "Progetto MIUR PRIN CINA Prot. 2010LHT4KM".

BICT 2014, December 01-03, Boston, United States

DOI 10.4108/icst.bict.2014.257976

$$
\begin{array}{rll}
P, Q ::= & \mathbf{0} & \text{Empty Process} \\
| & X(v) & \text{Entity Instance} \\
| & P \mid Q & \text{Parallel Composition} \\
| & (\nu a^{n,rad}).P & \text{Restriction} \\
\pi ::= & \mathtt{delay}^{p,n,\mathrm{U}} & \text{Delay} \\
| & !u(v)^{\mathrm{U}} & \text{Output} \\
| & ?u(x)^{p,\mathrm{U}} & \text{Input} \\
| & \mathtt{mov}^{\mathrm{U}} & \text{Move} \\
M ::= & \pi.P \mid \pi.P + M & \text{Choice} \\
N ::= & M \mid (\nu a^{n,rad}).N & \text{Restricted Choice} \\
u ::= & a \mid b \mid \cdots \mid x \mid y \mid \cdots & \text{Identifiers} \\
v, w ::= & u \mid \mathtt{c} \mid \mathtt{x} \mid \mathtt{y} \mid \ldots & \text{Expressions} \\
D ::= & \emptyset \mid D, X(x) = N^{\xi,\omega,\sigma} & \text{Entity Definitions} \\
E ::= & \emptyset \mid E, a^{n,rad} & \text{Channel Declarations} \\
W ::= & X(v) \mid \ldots \mid X(v) & \text{World} \\
C ::= & \mathtt{when}\ R\ \mathtt{run}^{\mathrm{U}}\ X(v) \ldots X(v) & \text{Commands} \\
| & C; C \mid \mathtt{skip} &
\end{array}
$$

Figure 2: Syntax of BIOSCAPE^P

- $\mathrm{U} \circ V(\mathtt{x}) = V(\mathtt{x}) + k$ *if* $\mathrm{U}(\mathtt{x}) = k$,
- $\mathrm{U} \circ V(\mathtt{x}) = V(\mathtt{x})$ *if* $\mathtt{x} \notin dom(\mathrm{U})$

The *conditional command* $\mathtt{when}\ R\ \mathtt{run}^{\mathrm{U}}\ X(v) \ldots X(v)$ adds new agents to the system when R is true. R is a decidable *predicate over the variables of the global state*. Given a predicate R and a state V, $R(V)$ is the result of evaluating the predicate R after substituting the global variables occurring in R with their values in V. Furthermore, if R is true, the global state is updated with U. Sequences of conditional commands can be used to repeat an event at given time intervals, such as in the case of drug dosage administration, or as often as condition R is satisfied. Consider the example in Fig. 3, where $\mathtt{clock} = 0$ is satisfied only once, but $\mathtt{nBac}{>}1$ and $\mathtt{clock}|10$ may be true more than once. Furthermore, repetitions are limited by the value of $\mathtt{count}$, a user-defined variable, and only at times divisible by 10.

$$
\begin{array}{l}
\mathtt{when}\ (\mathtt{clock} = 0)\ \mathtt{run}^{[\mathtt{count:1}]}\ \mathtt{Bac}_1 \ldots \mathtt{Bac}_n\ \mathtt{AmA}_1 \ldots \mathtt{AmA}_k; \\
\mathtt{when}\ (\mathtt{nBac}{>}1 \wedge \mathtt{count} \leq 6 \wedge \mathtt{clock}|10) \\
\qquad \mathtt{run}^{[\mathtt{count:1}]}\ \mathtt{AmA}_1 \ldots \mathtt{AmA}_k;
\end{array}
$$

Figure 3: Antimicrobial agents are added up to 6 times while there are bacteria in the system and periodically when the clock is divisible by 10.

$C; C$ represents a sequence of commands, and $\mathtt{skip}$ is the do nothing command to halt execution.

The language describing the entities behavior is a π-calculus in which we have a non-deterministic and a probabilistic choice, inspired by the calculi introduced in [7], [6], and [13].

We assume a set of *channel names*, denoted by a, b, a set of *variables*, denoted by x, y, and *constants for real numbers*, denoted by $\mathtt{c}$, with subscripts or superscripts, if needed. The empty process is $\mathbf{0}$. By $X(v)$ we denote an instance of the entity defined by X. The process $P \mid Q$ is the parallel composition of processes P and Q. By $(\nu a^{n,rad}).P$ we define the channel name a with two parameters n and *rad*$\in \mathbb{R}_{\geq 0}$. The radius is the maximum distance between processes in order to communicate through channel a; n determines how long it takes for two processes to react given that they are close enough to communicate.

The *heterogeneous* choice is denoted by M. Choices may have reaction branches and movement branches. As we will see reactions are probabilistic whereas movement is non-deterministic. The restricted choice, denoted by N, is a choice of prefixed processes M with top level local channel definitions. The prefix π denotes the action that the process $\pi.P$ can perform. The prefix $\mathtt{delay}^{p,n,\mathrm{U}}$ is a spontaneous and unilateral reaction of a single process, where $p \in (0, 1]$ represents a positive probability, n is the time it takes for P to become available, and the superscript U specifies the changes of the state generated by the reaction. The prefix $!u$ denotes output, and the prefix $?u$ denotes input. Output on a channel specifies an expression whose value will be sent, and input on a channel specifies a variable, that will be bound to the value received. Input and output prefixes, $!u(v)^{\mathrm{U}}$ and $?u(x)^{p,\mathrm{U}}$, have a user defined update function U on state V. Moreover, the input prefix has an associated probability p, whereas, the output one does not. As in [7], we consider inputs as generating the action, and output as reacting. The prefix $\mathtt{mov}$ moves processes in space according to their diffusion rate (ω) (see below). As it is the case for the other prefixes, $\mathtt{mov}$ has an update function U associated with it. The superscript U is omitted, when there is no update of the state associated with the action. We use standard syntactic abbreviations such as π for $\pi.0$. The input prefix $?u(x)$ and the restriction νa^- are binders, and they define the scope of the variable x and the channel name a respectively.

Expressions may be channel names, variables, constants, and variables $\mathtt{x}, \mathtt{y}$ in the global state V.

We use E to range over environments of channel name declarations. A channel name a appears at most once in E, so we consider E up to permutations of channel declarations. The *domain of* E ($\mathtt{dom}(E)$) is the set of channel names declared in E.

D is a list of entity definitions. The entities appearing in W and in C need to be specified in D. The domain of D ($\mathtt{dom}(D)$) is the set of entity names declared in D. An entity definition has the form $X(x) = N^{\xi,\omega,\sigma}$, and it defines entity X, with formal parameter x, to be the restricted choice N with geometry ξ, ω, σ, specifying a movement space ξ, a step ω, and a shape σ. Multiple parameters could be defined with the richer expression language of [5], which includes tuples. The restricted choice $N = \nu_1 \ldots \nu_n.M$, where ν_i, $1 \leq i \leq n$, is an abbreviation for $\nu a_i^{n_i,rad_i}$, describes the behavior of X. The a_i are the private channels of the entity. Entities whose definition N has either delay or input branches are called *reacting entities*. For a reacting entity, *the sum of the probabilities on the delay and input branches is equal to 1*. The selection of one of the choices is probabilistic, and it depends not only on the available interactions with other processes, but also on the available space. The movement space ξ is a shape containing the entity shape that defines the limit of the possible movements of the entity, so that X can move within ξ. The step $\omega \in \mathbb{R}_{\geq 0}$ is the distance that X can move in a unit of time, and it corresponds to the diffusion rate of X; σ is the three-dimensional shape (sphere, cube, etc.) of X, having a barycentre. The movement space for the empty process $\mathbf{0}$ is everywhere, the global space, and its movement step is 0. Each entity variable X can be defined at most once in D, and the only free variable of N is x. The global state V also contains population information of entities defined in D.

2.1 Example of BIOSCAPE^P modeling

A model of the interaction between bacteria and antimicrobial agents, is given by the example program in Fig.4.

```
 1.  val orig=<1.0, 2.0, 3.0>
 2.  val bactArea=cuboid(100.0,100.0,100.0)@orig
 3.  val step=0.1
 4.  new kill@1,1.0
 5.  let Bac()@bactArea,step,sphere(1.0)=
 6.    do delay@0.5,5,[pH:k];(Bac()|Bac())
 7.    or ?kill()@0.5;BacDead()
 8.    or mov;Bac()
 9.  and BacDead()@bactArea,0.0,sphere(1.0)=
10.    do delay@1.0,1;()
11.  and AmA()@bactArea,step,sphere(0.5)=
12.    do !kill();()
13.    or mov;AmA()
14.  when (clock = 0) run^[count:1] Bac_1 ... Bac_n AmA_1 ... AmA_k;
15.  when (nBac>1∧clock|10) run^[count:1] AmA_1 ... AmA_k;
```

Figure 4: BIOSCAPE^P model of bacteria–antimicrobial interactions

Antimicrobial agents are represented by the entity AmA. To model the the fact that bacteria, once they get in contact with AmA die, we define two entities: Bac and BacDead, representing bacteria and dead bacteria respectively. Both bacteria and antimicrobial agents are confined in the area defined by the shape bactArea. Moreover, Bac and AmA move in this area with step=0.1, whereas a BacDead does not move. The behavior of entities is defined by the do statement on the right of the equal sing. The do or ... construct is the concrete syntax for the discriminated choice, also semicolon is used instead of colon after the prefix.

We assume the existence of two global variables pH and count, representing the pH of the media and the number of doses of antibiotic delivered, respectively.

A bacterium, say $\texttt{Bac}_1$, may divide and be replaced by two new bacteria, $\texttt{Bac}_2$, and $\texttt{Bac}_3$, see line 6 of the code. The delay prefix has associated a probability of 0.5 and the parameter 5 indicates the duration of the reaction. This reaction updates pH by k, representing the acidification of the system with bacterial growth ([pH:k]). Alternatively, in line 7, a bacterium may synchronize on channel kill with an AmA entity, $\texttt{AmA}_1$, in line 12 of the code. Again the probability of this reaction is 0.5. The result of this synchronization replaces $\texttt{Bac}_1 \mid \texttt{AmA}_1$ with a dead bacterium, BacDead. This reaction, as specified by the definition of the channel in line 4, has duration 1, and may happen only if $\texttt{AmA}_1$ and $\texttt{Bac}_1$ are closer than the radius of channel kill, i.e., 1.0. Line 8, says that a bacterium is subject to Brownian motion. This is simulated by moving bacteria the distance specified by step=0.1, at every clock cycle. Observe that movement causes no change to the global state, and it is not part of the probabilistic behavior. Line 10 specifies that a dead bacterium, BacDead, will disappear after 1 clock cycle with probability 1.0. This is its only possible action. Line 11 defines an antimicrobial agent, AmA, which can either kill a bacterium and disappear (line 12) in one clock cycle, or move (line 13). Lines 14 and 15 contain the simulation command, where each conditional command increases the number of doses of antimicrobial agents by 1 ([count:1]).

3. OPERATIONAL SEMANTICS

A novel aspect of BIOSCAPE^P is the ability to describe complex experiments, beyond specifying initial concentrations. The user describes an experiment by designing a simulation command C. See Figs. 1 and 3 for examples. These new commands need to be evaluated at the beginning of simulation, and also during execution, implying the need to interleave reaction simulation and command execution. In order to achieve that, we propose a multi-level semantics for BIOSCAPE^P. On the one hand, we have entities that can react either by evolution (delay), or by interaction (send/receive and proximity), and entities can also move (mov); we call this behavior *Individual Level Semantics*. On the other hand, the world can evolve by events triggered by simulation commands altering the world instantaneously, by adding entities, which we call *World Level Semantics*. As we will see, the individual semantics modifies the simulation clock, while the world level semantics might execute commands depending on the simulation clock without altering it. However, except for the simulation clock, both the Individual and the World Level semantics may modify the global state.

3.1 World Level Semantics

The execution of commands and the interleaving with individual and world semantics appear in Fig. 5. The main function is $\text{SIM}_D^{E_0}(V, F, C)$, which executes a simulation step. The function is parametric on the definition of entities of the system, D, and the set of global channels, E_0. For the example of Fig. 4, D contains the definition of the three entities Bac, BacDead, and AmA, and E_0 contains the declaration of channel kill. The inputs of the function are: V, the global state of variables; F, the run-time configuration, that shows the entities of the system. In the run-time configuration, which is formally defined in Definition 3.6, we track the evolution of entities and their locations in space over time; C, the simulation command that needs to be re-evaluated at each step. The output is the pair: "global state, run-time configuration", returned by the execution of the individual semantics INDSEM_D on: global state V' as modified by the evaluation of the simulation command, and run-time configuration F' obtained by placing in space the entities in W which are added to the system by the execution of the simulation command C.

The global channels in E_0 and the definitions in D are needed to run the individual semantics.

The function $\text{EVAL}(V, C)$ returns a pair V', W, where W contains the entities from conditional commands in C whose guards are satisfied, and V' is obtained by applying the update functions of those same conditional commands to V.

The function $\text{PLACE}(W, F)$ adds the entities in W to a run-time configuration F. The entities in W will be positioned randomly in the available space. The formal definition of this function is given in Definition 3.9. The random positioning of entities in space is like in BIOSCAPE[4], but unlike in BIOSCAPE^L[5], where entity locations are programmable and therefore specified by the user.

$\text{INDSEM}_D(E, V, F)$ will be defined in the next section.

We can now define a simulation.

DEFINITION 3.1 (SIMULATION). *A simulation is a sequence* $\text{SIM}_D^{E_0}(V_0, F_0, C) \to \text{SIM}_D^{E_0}(V_1, F_1, C) \to \text{SIM}_D^{E_0}(V_2, F_2, C) \to \cdots$ *where* $\text{SIM}_D^{E_0}(V_i, F_i, C) = V_{i+1}, F_{i+1}$. *Finally,* $\text{INIT}(D, E_0, V, C)$ *starts a simulation on the empty run-time configuration.*

3.2 Individual Semantics

The individual semantics has three components. The first component consists of entity reactions such as input/output interaction and spontaneous delay. In the example of Fig. 4, Bac|AmA can communicate with probability 0.5 through ?kill/!kill producing BacDead. In doing so, the global state is automatically updated, where nAmA and nBac are decremented by 1, and nBacDead

$$\textsc{eval}(V, \texttt{skip}) = V, \mathbf{0} \qquad \text{(C.Skip)}$$

$$\textsc{eval}(V, \texttt{when } R \texttt{ run}^{\mathfrak{u}} \overline{X(v)}) = \mathfrak{u} \circ V, X_1(v_1) \mid \cdots \mid X_n(v_n) \quad \text{if } R(V) \qquad \text{(C.IfTr)}$$

$$\textsc{eval}(V, \texttt{when } R \texttt{ run}^{\mathfrak{u}} \overline{X(v)}) = V, \mathbf{0} \quad \text{if } \neg R(V) \qquad \text{(C.IfFs)}$$

$$\textsc{eval}(V, C; C') = V'', W \mid W' \qquad \text{(C.Seq)}$$
where $\textsc{eval}(V, C) = V', W$
and $\textsc{eval}(V', C') = V'', W'$

$$\textsc{sim}_D^{E_0}(V, F, C) = \textsc{indSem}_D(E_0, V', F') \qquad \text{(SimStep)}$$
where $\textsc{eval}(V, C) = V', W$
and $\textsc{place}(W, F) = F'$

$$\textsc{init}(D, E_0, V, C) = \textsc{sim}_D^{E_0}(V, \mathbf{0}, C) \qquad \text{(Start)}$$

Figure 5: World Level Semantics

is incremented by 1. `Bac` can also evolve into `Bac|Bac` through `delay`. In this step there is an implicit update where `nBac` is incremented by 1, and an explicit update specified by the programmer, where `pH` is acidified by k. The second component consists of movement. Entities located in space might be able to move. In our example `Bac` and `AmA` move at the same speed determined by `step`, but `BacDead` does not move. An `AmA` at location pt is located a step away after executing `mov`. The third component of the individual semantics is time. The clock advances one tic at a time, and the timer on timed processes is decremented bringing them closer to completion. At any given time there may be many reactions available, and, because of the non-determinism of the system, there are many ways to choose a maximal number of reactions to execute. For instance, if the internal probabilistic choice of a bacterium selects an input on channel `kill`, and there are 10 `AmA`, we have 10 pairs of `?kill/!kill`, and of these pairs more than one in entities close enough to be able to react.

We call $\textsc{prreact}(E, V, F)$ the result of executing a maximal number of available reactions in F, with global state V, and channels E. We also call $\textsc{prmove}(V, F)$ the result of moving all entities in F. Finally we call $\textsc{prtime}(V, F)$ the result of updating the clock in V and updating the timer on all timed entities in F. The individual semantics consists of composing these three parallel operations: $\textsc{indSem}_D(E, V, F) = \textsc{prtime}(\textsc{prmove}(\textsc{prreact}(E, V, F)))$

Consider this example. A biologist is performing an experiment in her lab where she is studying the effect of antibiotics on a population of bacteria. She sets up her experiment and in six hours she needs to add a new dose of antibiotics. She sets up a timer for six hours while her initial concentrations are reacting. The reactions of the initial concentrations during the initial six hours are captured by process $\textsc{indSem}_D$. In this example there are two conditional commands, the first one satisfied only at time t_0 where the initial concentrations are set. The next conditional command is to add the new dose of antibiotics when the timer rings. When the timer rings, the condition *time*=t_0 + 6*hs* becomes true, and it is handled by rule $\textsc{sim}_D^{E_0}$, where the current agents, which may have more or less bacteria than at initial time t_0 and will most likely have less antibiotic, are boosted with a new dose of antibiotics.

The rest of the section is devoted to the formalization of these three components of the individual semantics. Fig. 6 defines when two configurations are structurally equivalent; Fig 7 contains the rules for one-step send/receive and `delay` reactions; Fig. 8 defines one-step movement, and Fig. 11 formalizes the parallel reduction relations PRReact, PRMove, and PRTime.

Unlike its predecessors: BioScape, Parallel BioScape, and BioScape^L, our new language does not have stochastic reactions, but a probabilistic choice of the actions that an entity may perform. For reactions we specify the time taken to reach completion.The other source of uncertainty is movement.

In order to combine the state modifications incurred during parallel reductions, we define state update composition as follows.

Definition 3.2 (Composition of Updates).

(i) Let $\mathfrak{u}_1$ *and* $\mathfrak{u}_2$ *be state updates,* $\mathfrak{u}_1 \bigoplus \mathfrak{u}_2$ *is defined by:*

- $\mathfrak{u}_1 \bigoplus \mathfrak{u}_2(\mathsf{x}) = \mathfrak{u}_1(\mathsf{x}) + \mathfrak{u}_2(\mathsf{x})$ *if* $\mathsf{x} \in dom(\mathfrak{u}_1) \cap dom(\mathfrak{u}_2)$,
- $\mathfrak{u}_1 \bigoplus \mathfrak{u}_2(\mathsf{x}) = \mathfrak{u}_i(\mathsf{x})$ *if* $\mathsf{x} \in dom(\mathfrak{u}_i)$ $(1 \leq i \leq 2)$*, and*
- $\mathfrak{u}_1 \bigoplus \mathfrak{u}_2(\mathsf{x})$ *undefined if* $\mathsf{x} \notin dom(\mathfrak{u}_1) \cup dom(\mathfrak{u}_2)$.

(ii) $\bigoplus_{i=1}^{m} \mathfrak{u}_i = \mathfrak{u}_1 \oplus \cdots \oplus \mathfrak{u}_m$

Commutativity and associativity of updates can be derived from commutativity and associativity of addition on $\mathbb{Z}$. So we have the following Lemma.

Lemma 3.3 (Commutativity of Update). *Let p be a permutation over* $\{1 \cdots n\}$, $\bigoplus_{i=1}^{m} \mathfrak{u}_i = \bigoplus_{i=1}^{m} \mathfrak{u}_{p(i)}$.

Notice that a user defined update function $\mathfrak{u}$ can never update the experimental clock. Simulation steps are the only way to update `clock` (See rule PR.Time in Fig. 11). Each conditional command and each `delay`, `mov`, and input/output prefix has an update function $\mathfrak{u}$ associated with it. The update is meant to happen when the prefix gets evaluated.

$$\text{S.Loc}\ \frac{P \equiv Q}{\{P\}_\mu \equiv \{Q\}_\mu} \qquad \text{S.Loc.Nu}\ \frac{}{(\nu a^{\mathtt{n,rad}}).\{P\}_\mu \equiv \{(\nu a^{\mathtt{n,rad}}).P\}_\mu}$$

$$\text{S.Loc.Par}\ \frac{\mu_1(\mathtt{shape}(P)) \cup \mu_2(\mathtt{shape}(Q)) = \mu(\mathtt{shape}(P \mid Q))}{\{P\}_{\mu_1} \mid \{Q\}_{\mu_2} \equiv \{P \mid Q\}_\mu}$$

$$\text{S.Nu.Com}\ \frac{}{(\nu a^{\mathtt{n,rad}}).(\nu b^{\mathtt{n'},\mathtt{rad'}}).A \equiv (\nu b^{\mathtt{n'},\mathtt{rad'}}).(\nu a^{\mathtt{n,rad}}).A}$$

$$\text{S.Nu.Par}\ \frac{a \notin \mathtt{fn}(B)}{((\nu a^{\mathtt{n,rad}}).A) \mid B \equiv (\nu a^{\mathtt{n,rad}}).(A \mid B)}$$

Figure 6: Structural Equivalence of Spatial Configurations

To define the run-time configuration of the system we start by adding to the description of entities of the system, an abstract notion of positioning into space. To this extent we use the map μ, to denote barycentre, orientation, and (possible) resizing of the shape of entities. Spatial configurations are defined as follows.

Definition 3.4 (Spatial Configurations). Spatial configurations*, denoted by A, B, . . . are defined as follows:*

$$A, B ::= \{P\}_\mu \quad | \quad A \mid B \quad | \quad (\nu a^{n,rad}).A$$

Processes in space may share communication channels.

Entity instances, $\{X(v)\}_\mu$, are called *located entities*. The volume occupied by a parallel composition of entities, is derived from the shape of its individual components, as follows.

RR.DELAY

$$\frac{X(x) = (\nu_1 \ldots \nu_n.(\texttt{delay}^{p,\mathrm{n},\mathrm{U}}.P\ [+\ M]))^{\xi,\omega,\sigma} \in D \qquad p_s = f(p, S[\{X(v)\}_\mu])}{E \vdash V, S[\{X(v)\}_\mu] \xrightarrow{p_s,\mathrm{n}} \mathrm{u} \circ V, S[\nu_1 \ldots \nu_n.\{P[v/x]\}_\mu]}$$

RR.COM

$$\frac{\begin{array}{c} Y(y) = (\nu'_1 \ldots \nu'_m.(!a(v)^{\mathrm{U}_1}.P\ [+\ M]))^{\xi',\omega',\sigma'} \in D \\ X(x) = (\nu_1 \ldots \nu_n.(?a(z)^{p,\mathrm{U}_2}.Q\ [+\ N]))^{\xi,\omega,\sigma} \in D \\ \texttt{dis}(\mu(\sigma), \mu'(\sigma')) \leq \texttt{rad} \qquad p_s = f(p, S[\{X(v_x)\}_\mu \mid \{Y(v_y)\}_{\mu'}]) \end{array}}{\begin{array}{c} E, a^{\mathrm{n},\mathrm{rad}} \vdash V, S[\{X(v_x)\}_\mu \mid \{Y(v_y)\}_{\mu'}] \xrightarrow{p_s,\mathrm{n}} (\mathrm{u}_1 \bigoplus \mathrm{u}_2) \circ V, \\ S[\nu_1 \ldots \nu_n.\{Q[v_x/x][v/z]\}_\mu \mid \nu'_1 \ldots \nu'_m.\{P[v_y/y]\}_{\mu'}] \end{array}}$$

RR.STR

$$\frac{A \equiv A' \qquad E \vdash V, A' \xrightarrow{p,\mathrm{n}} V', B' \qquad B' \equiv B}{E \vdash V, A \xrightarrow{p,\mathrm{n}} V', B}$$

Figure 7: Reaction Relation

MR.MOVE

$$\frac{\mu' = \texttt{translate}(\omega, \mu) \quad \mu'(\sigma) \subseteq \xi \qquad X(x) = (\nu_1 \ldots \nu_n.(\texttt{mov}^{\mathrm{U}}.P\ [+\ M]))^{\xi,\omega,\sigma} \in D}{V, \{X(v)\}_\mu \rightarrow \mathrm{u} \circ V, \nu_1 \ldots \nu_n.\{P[v/x]\}_{\mu'}}$$

MR.STR

$$\frac{A \equiv A' \qquad V, A' \rightarrow V', B' \qquad B' \equiv B}{V, A \rightarrow V', B}$$

Figure 8: Motion Relation

DEFINITION 3.5 (SHAPE).

$$\begin{array}{l} \mathit{shape}(\mathbf{0}) = \emptyset \\ \mathit{shape}(X(v)) = \sigma \ \ \mathit{if}\ X(x) = M^{\xi,\omega,\sigma} \in D \\ \mathit{shape}((\nu a^{\mathrm{n},\mathrm{rad}}).P) = \mathit{shape}(P) \\ \mathit{shape}(P \mid Q) = \mathit{shape}(P) \Cup \mathit{shape}(Q) \end{array}$$

where $\Cup$ gives a shape obtained by composing two shapes trough juxtaposition. For different applications we can choose suitable functions to realize $\Cup$, we only require $\Cup$ to be a commutative and associative operator, i.e. $\sigma_1 \Cup \sigma_2 = \sigma_2 \Cup \sigma_1$ and $(\sigma_1 \Cup \sigma_2) \Cup \sigma_3 = \sigma_1 \Cup (\sigma_2 \Cup \sigma_3)$.

$\mu(\texttt{shape}(P))$ denotes the space, that is the set of points, occupied by a process P in the global coordinate system.

Structural equivalence on spatial configurations is defined in Fig. 6, omitting the rules for associativity and commutativity of $\mid$ and $+$. Rule S.LOC uses the standard structural equivalence of Pi-calculus processes. We assume that structural equivalence on spatial configurations is also such that parallel composition is commutative, associative, and has neutral element $\{0\}_\mu$ for any μ. The premise of rule S.LOC.PAR assures that the two equivalent processes occupy exactly the same space. In rule S.NU.PAR, fn is a function that returns the set of free channel names of a configuration.

The (parallel) operational semantics of BIOSCAPE^P is based on two *underlying* reduction relations: a reaction relation, $E \vdash V, A \xrightarrow{p,\mathrm{n}} V', B$, for reactions such as synchronization and delay, defined in Fig. 7, and a motion relation, $V, A \rightarrow V', B$, for geometric transformations, in our case movements, defined in Fig. 8. Notice that reduction axioms (RR.DELAY, RR.COM, MR.MOVE) only involve entities ($X(v)$), and entities evolve according to one of the choices in their definition.

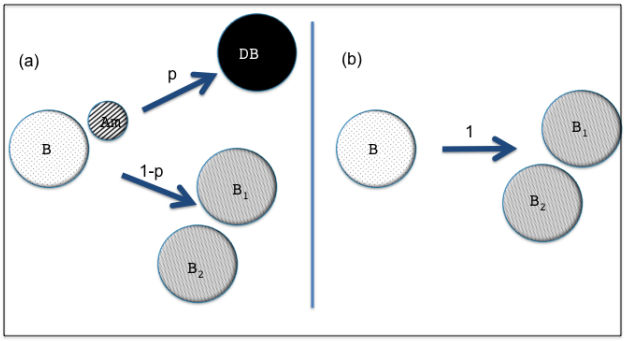

Figure 9: Probabilities and 3D Space

Fig. 7 defines the reaction reduction relation of BIOSCAPE^P, $E \vdash V, A \xrightarrow{p,\mathrm{n}} V', B$, where p is the probability of the reduction and n is the duration of the communication or delay. We write $\texttt{dis}(\mu(\sigma), \mu'(\sigma'))$ for the distance between the shapes, σ, and σ', positioned in the global frame by μ, and μ'. The potential overhead of computing the distance between entities in order to establish if they can react could be mitigated by partitioning entities in groups based on their location. In rule RR.COM the condition $\texttt{dis}(\mu(\sigma), \mu'(\sigma')) \leq \texttt{rad}$ ensures that located entities $\{X(v_x)\}_\mu$ and $\{Y(v_y)\}_{\mu'}$ are close enough to communicate through channel a.

In rules RR.DELAY, and RR.COM, $S[\_]$ denotes a *spatial context* containing the parallel composition of entities that, given the position of the located entity $\{X(\_)\}_\mu$, may react with it. This context is used to identify such entities, and, as in [7], normalize the probabilities.

Fig. 9 helps us understand probabilistic choice in a 3D environment. Assume that AmA and Bac are as in Fig. 4, in particular, observe that Bac can either be killed by an antimicrobial agent or divide into two. In Fig. 9(a), since there are a Bac and an AmA close enough to react, Bac can either be killed with probability p, or divide, with probability 1-p. However, in Fig. 9(b), there is only one possible evolution for Bac, and therefore, its probability is 1. Instead, if there were two AmA's close enough to interact with Bac, the probability of dividing would be 1-2p. This simple example highlights the fact that probabilities are balanced according to the available reactions in the current configuration. To this extent, when executing reactions with RR.DELAY and RR.COM, we recompute using the function f the probability of the reaction, taking into account the original probability p, and the entities surrounding the reacting entity $\{X(\_)\}_\mu$.

The motion reduction relation of BIOSCAPE^P, $V, A \rightarrow V', B$, is defined in Fig. 8. By $\texttt{translate}(\omega,\mu)$ we denote the function that randomly generates a new map μ', using the movement step ω and the old map μ. The condition $\mu'(\sigma) \subseteq \xi$ of rule MR.MOVE ensures the new located process $\{P[v/x]\}_{\mu'}$ is within the movement space ξ of X.

Since reductions may have different durations, we use *timed configurations*, $\{\!\{A\}\!\}^n$, meaning that, after a time n, this configuration will be the located process A as in [3].

DEFINITION 3.6 (SPATIO-TEMPORAL CONFIGURATIONS).

$$F, G ::= A \ \mid\ \{\!\{A\}\!\}^n \ \mid\ F \mid G \ \mid\ (\nu a^{n,rad}).F \qquad (n \geq 0)$$

We extend structural equivalence to timed configurations in Fig. 10.

$$\text{S.TI}\ \frac{A \equiv B}{\{\!|A|\!\}^n \equiv \{\!|B|\!\}^n} \qquad \text{S.TI.ZERO}\ \frac{}{\{\!|A|\!\}^0 \equiv A}$$

Figure 10: Structural Equivalence of Timed Configurations

BIOSCAPE^P has been designed to be executed in a distributed environment. For this purpose, in the following we formalize a parallel semantics for the language. To define the parallel reductions, PRREACT, PRMOVE, and PRTIME, where more than one redex is reduced in a single step. we introduce multi-hole contexts C by $C ::= F \mid [\,] \mid C \mid C \mid (\nu x^{n,\mathtt{rad}}).C$

Congruence on multi-hole contexts is naturally induced by the congruence on configuration, associativity and commutativity of the parallel operator, and standard rules for ν restrictions similar to S.NU.COM and S.NU.PAR. Given this congruence any multi-hole context, C, may be written in a *canonical form*. That is, there is $C', C \equiv C'$ such that $C' = \nu_1 \ldots \nu_n.F_1 \mid \cdots \mid F_m \mid [\,] \mid \cdots \mid [\,]$ and for all j, $1 \leq j \leq m$, $F_j = \{\!|A|\!\}^m$ for some A, and m, or $F_j = \{X(\overline{v})\}_\mu$ for some P, and μ. We say that $a_1^{n_1,rad_1}$, ..., $a_m^{n_m,rad_m}$ is $\mathtt{restr}(C)$. In the following we assume that multi-hole contexts are always in canonical form.

Our reduction strategy avoids spatial clashes. In particular for moving reductions we have to ensure that moves and reshaping are compatible with the available space, that is after moving no entity overlaps another entity. For reaction reductions we have to assure that the created entities have their space. To this aim we define the space of a configuration, and a predicate that says whether a configuration does not have any overlapping entities.

DEFINITION 3.7 (SPACE). *Let $\mathtt{space}(F)$ be a function on configuration F that returns the space occupied by its processes located in the global frame defined as follows.*

$$\begin{array}{ll}
\mathtt{space}(\{P\}_\mu) & = \mu(\mathtt{shape}(P)) \\
\mathtt{space}(\{\!|A|\!\}^n) & = \mathtt{space}(A) \\
\mathtt{space}(F \mid G) & = \mathtt{space}(F) \cup \mathtt{space}(G) \\
\mathtt{space}((\nu a^{n,rad}).F) & = \mathtt{space}(F)
\end{array}$$

We say that a configuration F is ok if no two entities in F overlap.

DEFINITION 3.8 (ok CONFIGURATION).

$\{P\}_\mu$ ok

If A ok *then* $\{\!|A|\!\}^n$ ok

If F ok *then* $(\nu x^{n,rad}).F$ ok

If F ok, G ok, *and* $\mathtt{space}(F) \cap \mathtt{space}(G) = \emptyset$ *then* $F \mid G$ ok

DEFINITION 3.9. *Given a configuration F, and a world W, define,* PLACE(W, F) *by:*

- PLACE$(\mathbf{0}, F) = F$,
- PLACE$(X(v) \mid W, F) = F' \mid \{X(v)\}_\mu$, *where* PLACE$(W, F) = F'$, *and the map μ is randomly generated such that $F' \mid \{X(v)\}_\mu$ is* ok.

LEMMA 3.10. *For all W, if F is* ok *and $F' =$* PLACE(W, F), *then F' is* ok.

Using the notion of ok configuration we define two notions of well-formedness of configurations. The first to be used for parallel reaction reductions and the second for parallel move reductions. We say that a configuration obtained by a set of parallel moves is ok_{mv} if it is ok and any "extra" movement would produce a clash, i.e. a configuration where some entities occupy the same space. Similarly we say that a configuration obtained by a set of reaction transformations is ok_r if it is ok and any "extra" transformation would produce a clash.

PR.REACT

$$\frac{E, \mathtt{restr}(C) \vdash V, S_i[A_i] \xrightarrow{p_i, \mathrm{n}_i} (\mathrm{U}_i \circ V), S_i[B_i] \quad 1 \leq i \leq m \qquad C[S_1[B_1]] \cdots [S_m[B_m]]\ \mathrm{ok}_r}{E \vdash V, C[S_1[A_1]] \cdots [S_m[A_m]] \dashrightarrow \bigoplus_{i=1}^{m} \mathrm{U}_i \circ V, C[S_1[\{\!|B_1|\!\}^{n_1}] \cdots S_m[\{\!|B_m|\!\}^{n_m}]]}$$

PR.MOVE

$$\frac{V, F_i \to V_i, G_i \quad V_i = \mathrm{U}_i \circ V \quad (1 \leq i \leq m) \qquad C[G_1] \cdots [G_p]\ \mathrm{ok}_{mv}}{V, C[F_1] \cdots [F_p] \rightarrowtail \bigoplus_{i=1}^{m} \mathrm{U}_i \circ V, C[G_1] \cdots [G_p]}$$

PR.TIME

$$\frac{C \text{ does not contain timed configurations}}{V, C[\{\!|A_1|\!\}^{n_1}] \cdots [\{\!|A_p|\!\}^{n_p}] \rightsquigarrow [\mathtt{clock} : 1] \circ V, C[\{\!|A_1|\!\}^{n_1 - 1}] \cdots [\{\!|A_p|\!\}^{n_p - 1}]}$$

PR.INDSEM

$$\frac{E \vdash V, F \dashrightarrow V_1, F_1 \rightarrowtail V_2, F_2 \rightsquigarrow V', F'}{E \vdash V, F \xrightarrow{D} V', F'}$$

Figure 11: Parallel Reduction Relation

We are now able to explain our parallel reduction strategy, whose rules are given in Fig. 11. The first three rules deal respectively with parallel reaction, motion, and timed reductions, while the fourth rule maps configurations into configurations by first reducing a maximal number of reactions, then reducing all movements, and finally updating the simulation clock and all timed entities.

The condition of obtaining an ok_{mv} extended configuration in rule PR.MOVE assures that all possible moves in $C[F_1] \cdots [F_p]$ which do not cause a clash have been reduced. Similar effect is produced by the conditions that the extended configuration is ok_r. Finally the fact that in PR.TIME the context does not contain timed configurations says that we have decreased the counter of all the timed entities.

LEMMA 3.11. *Given entity definitions D, channel declarations E, global state V, and run-time configuration F. If F is* ok *and $E \vdash V, F \xrightarrow{D} V', F'$, then F' is* ok.

THEOREM 1. *Let* $\mathrm{SIM}_D^{E_0}(V, F, C) = V', F'$. *If F is* ok, *then F' is* ok.

PROOF. Let $F_1 =$ PLACE(W, F), where EVAL$(V, C) = V_1, W$ for some V_1. From the fact that F is ok and Lemma 3.10 we have that F_1 is ok. Let V', F' be such that $E_0 \vdash V_1, F_1 \xrightarrow{D} V', F'$, from Lemma 3.11, we have that F' is ok, which from definition of $\mathrm{SIM}_D^{E_0}$ in Fig. 5 proves the result. □

Given a BIOSCAPE^P model of an experiment, let E_0 be the declaration of channel common to all the entities, D be the entities definition, C the simulation command, and V_0 the set of initial values of global variables. Since the initial configuration of the system $\mathbf{0}$ is ok, we derive that at each step of the simulation we have ok configurations.

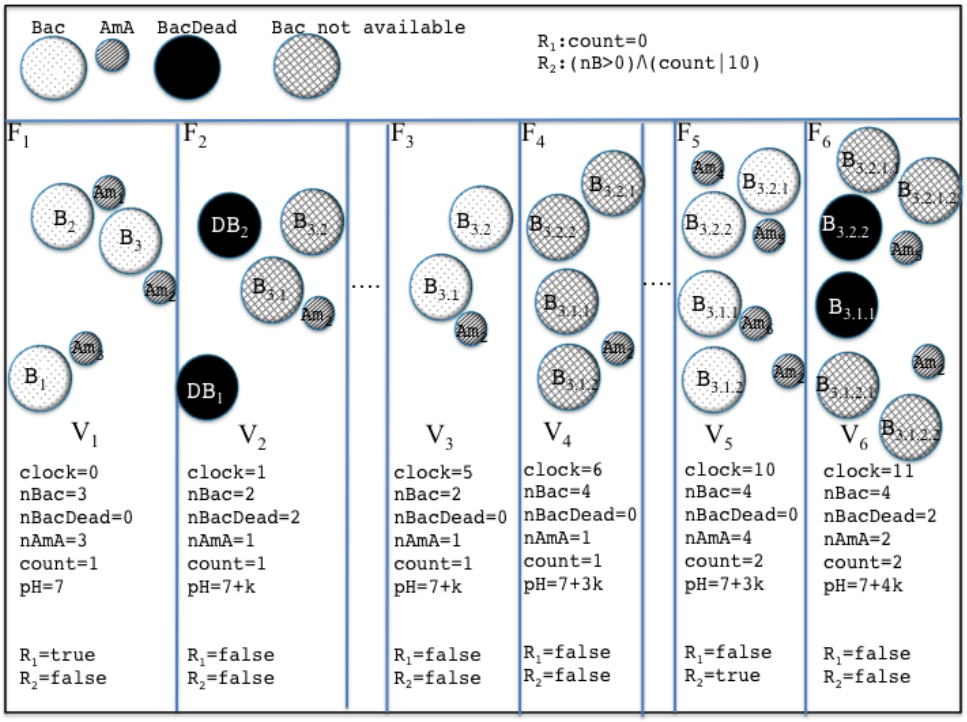

Figure 12: Example of two level operational semantics

3.3 Example of operational semantics

In Fig. 12 we show graphically an execution of the program in Fig. 4. We assume that C_1 is the command in line 14 with $n = k = 3$; C_2 is the command in line 15 with $k = 3$; V_0 is the initial global state in which, the variables `clock`, `nBac`, `nAmA`, `nBacDead`, and `count` have value 0 and `pH` 7; E_0, the global channels declaration is $\texttt{kill}^{1,1.0}$. The simulation starts with $\text{SIM}_D^{E_0}(V_0, \mathbf{0}, C_1; C_2)$. Since in V_0 the predicate of C_1 is true and the one of C_2 is false, we have $\text{EVAL}(V_0, C_1; C_2) = V_1, W$, where $W = \texttt{Bac} \mid \texttt{Bac} \mid \texttt{Bac} \mid \texttt{AmA} \mid \texttt{AmA} \mid \texttt{AmA}$, $V_1 = [\texttt{count}{:}1] \circ V_0$, and $\text{PLACE}(W, \mathbf{0}) = F_1$, where F_1 is $\{\texttt{Bac}\}_{c1} \mid \{\texttt{AmA}\}_{d3} \mid \{\texttt{Bac}\}_{c2} \mid \{\texttt{AmA}\}_{d1} \mid \{\texttt{Bac}\}_{c3} \mid \{\texttt{AmA}\}_{d2}$. The leftmost frame of Fig. 12 shows the runtime configuration F_1, and the global state V_1, which are the configuration and state after the execution of the world semantics, and before the execution of the individual semantics.

The following frame shows the result of the execution of the individual semantics $\text{INDSEM}_D(E_0, V_1, F_1)$ that is: $E_0 \vdash V_1, F_2 \xrightarrow{D} V_2, F_2$. We now describe how this is obtained.

Applying rule RR.COM with $S = [\,]$ we have

$E_0 \vdash V_1, S[\{\texttt{Bac}\}_{c1} \mid \{\texttt{AmA}\}_{d3}] \xrightarrow{0.5,1} V_1, S[A_1]$, and

$E_0 \vdash V_1, S[\{\texttt{Bac}\}_{c2} \mid \{\texttt{AmA}\}_{d1}] \xrightarrow{0.5,1} V_1, S[A_2]$, where $A_1 = \{\texttt{BacDead}\}_{c1} \mid \{\mathbf{0}\}_{d3}$, and $A_2 = \{\texttt{BacDead}\}_{c2} \mid \{\mathbf{0}\}_{d1}$, whereas applying rule RR.DELAY with $S' = \{\texttt{AmA}\}_{d2}$ we have

$E_0 \vdash V_1, S'[\{\texttt{Bac}\}_{c3}] \xrightarrow{0.5,5} [\texttt{pH}{:}k] \circ V_1, S'[A_3]$,

where $A_3 = \{\texttt{Bac}\}_{c4} \mid \{\texttt{Bac}\}_{c5}$.

Therefore, with the parallel rule PR.REACT we get

$$E_0 \vdash V_1, F_1 \dashrightarrow [\texttt{pH}{:}k] \circ V_1, \{\!\{A_1\}\!\}^1 \mid \{\!\{A_2\}\!\}^1 \mid \{\!\{A_3\}\!\}^5 \mid \{\texttt{AmA}\}_{d2}$$

The parallel rule PR.MOVE, applies rule MR.MOVE only to $\{\texttt{AmA}\}_{d2}$ which is the only entity not timed. Therefore, we have

$$V_1', F_1' \rightarrowtail V_1', \{\!\{A_1\}\!\}^1 \mid \{\!\{A_2\}\!\}^1 \mid \{\!\{A_3\}\!\}^4 \mid \{\texttt{AmA}\}_{d4}$$

where V_1', and F_1' are the state and configuration produced by PR.REACT. Finally, rule PR.TIME updates the clock and reduces the step that need to be done to complete the reactions:

$$V_1', F_2' \rightsquigarrow [\texttt{clock}{:}1] \circ V_1', \{\!\{A_1\}\!\}^0 \mid \{\!\{A_2\}\!\}^0 \mid \{\!\{A_3\}\!\}^4 \mid \{\texttt{AmA}\}_{d4}$$

Therefore $[\texttt{clock}{:}1] \circ V_1'$ is V_2 of Fig. 12, and the picture represents F_2 which is

$$\{\texttt{BacDead}\}_{c1} \mid \{\texttt{BacDead}\}_{c2} \mid \{\!\{\{\texttt{Bac}\}_{c4} \mid \{\texttt{Bac}\}_{c5}\}\!\}^4 \mid \{\texttt{AmA}\}_{d4}$$

In the following clock cycle, both $\{\texttt{BacDead}\}_{c1}$, and $\{\texttt{BacDead}\}_{c2}$ will become $\mathbf{0}$, and after four cycles $\{\!\{\{\texttt{Bac}\}_{c4} \mid \{\texttt{Bac}\}_{c5}\}\!\}^4$ will become $\{\!\{\{\texttt{Bac}\}_{c4} \mid \{\texttt{Bac}\}_{c5}\}\!\}^0$ which is congruent to $\{\texttt{Bac}\}_{c4} \mid \{\texttt{Bac}\}_{c5}$. This is shown in frame F_3.

In the subsequent frames we show a possible evolution of the experiment in which $\{\texttt{Bac}\}_{c4}$, and $\{\texttt{Bac}\}_{c5}$ divide. Note that, whereas $\{\texttt{Bac}\}_{c4}$ divide with probability 0.5, since there is $\{\texttt{AmA}\}_{d4}$ with which it could react, $\{\texttt{Bac}\}_{c5}$ divide with probability 1, since there are not `AmA`'s in its spatial context with which it could react. Till the penultimate frame the conditions of both commands C_1 and C_2 are false so there is no addition of entities. When $\texttt{clock} = 10$, since the number of `Bac` is greater than 0, the condition of command C_2 is true, so three new `AmA` are added, placed randomly in the space and the global variable `count` is updated by adding one. The result of the execution is shown in frame F_5. Finally, frame F_6 shows another possible evolution of the system, in which some of the bacteria react with the `AmA` agents, and others divide. The spatial contexts will partition the entities and the probability of the reactions will be recomputed accordingly. In particular, for the reaction of $\texttt{B}_{3.2.2}$ and $\texttt{Am}_4$ the spatial context is $[\,]$, so the probability of this reaction versus the division of $\texttt{B}_{3.2.2}$ is 0.5; for the division of $\texttt{B}_{3.2.1}$ the spatial context is $\texttt{Am}_5$, so the probability is 0.5; for the reaction of $\texttt{B}_{3.1.1}$ and $\texttt{Am}_6$ the spatial context is $[\,]$, and the probability is again 0.5, and for the division of $\texttt{B}_{3.1.2}$, assuming that $\texttt{Am}_2$ is not close enough to react, the spatial context is $[\,]$, and so the

probability is 1.0.

4. CONCLUSIONS AND RELATED WORK

BIOSCAPE [4] extends Priami's Stochastic Pi Calculus [10] with high-level primitives to describe stochastic processes in 3D space and introduces an abstract notion of movement to capture Brownian motion. BIOSCAPE was designed to program biological and biomaterial processes and their interactions. In BIOSCAPE, the programmer specifies entities at a higher level, unlike 3π [2], where the programmer has to specify spatial behavior using affine transformations. Compared with biological DSL, such as Cell Modeler [11], and GRO [8], in BIOSCAPE we can program a wider set of application. For instance, in BIOSCAPE^L [5], by introducing the notion of programmable locations, complex structures such as polymers, oligomers and complexes in 3D space can be modeled. Earlier work [3], defined a parallel semantics for BIOSCAPE, motivated by the desire to take advantage of the new multi-core and GPU architectures needed. The previous BIOSCAPE languages are based on a Markovian semantics using Gillespie's stochastic simulation algorithm. In contrast, our new probabilistic language, BIOSCAPE^P, takes full advantage of the three dimensional space by having a random move operation, and reaction on proximity. Instead of using reaction rates and populations to stochastically select reactions, it uses probabilities to choose among all possible reactions for each entity at any given time, and rebalancing probabilities as needed. The probabilistic branching is inspired by the Probabilistic Asynchronous π-Calculus [7]. We believe that this approach is more suitable in a distributed execution environment, since every computation unit could keep a set of spatially close entities, and communication between computational units could be limited to migration of entities through movement.

BIOSCAPE^P has been designed with a scalable semantics suitable for distributed computation. The commutativity of state update enables a parallel semantics and a multicore implementation where each parallel process can independently update the global state (Lemma 3.3 and Fig. 11). An implementation would take advantage of discrete event simulation queues to model updates of timed entities, so that instead of updating timers at each clock tick, entities will be released at the corresponding time.

A novel feature of BIOSCAPE^P is the `when-run` state dependent command. This new conditional command depends on a global state that includes the simulation clock, population of entities, and user defined variables. Execution of a command may introduce new entities in the system, thereby changing its topology. To the best of our knowledge, BIOSCAPE^P is the first modeling language with commands dependent on used defined global state.

In Bio-PEPA[1], the Heaviside function is used to switch a reactions on and off by periodically alternating its rate between 0 and non-zero. This is in contrast with our case, where instead of changing reaction rates, we add new agents to the system when a predicate is satisfied. Although Bio-PEPA does have access to a population information and the simulation clock, it can enable or disable reactions, and it can stochastically approximate our state-dependent conditional commands without capturing its deterministic behavior.

FM-Sim [12] is a Markovian domain-specific simulator for defining and simulating fluorescence microscopy assays. In FM-Sim, the user can define its experimental protocol in terms of scheduling events dependent on a global state. This dependence on state, enables the dynamically modification of reaction rates, in contrast to BIOSCAPE^P, where state dependency enables the infusion of new entities into the run-time system. To justify the consistency of the system, we define a notion of well-formed configuration (ok), and we show that our multi-level semantics preserves well-formedness.

To the best of our knowledge, BIOSCAPE^P is the first process algebra that captures the complexity of wet-lab experimentation by allowing the execution of conditional events while representing agent-level reactions.

Future work includes the implementation of a simulation tool for BIOSCAPE^P, and considering the possibility of using existing probabilistic model checkers, such as PRISM [9], to prove properties of the modeled system.

5. ACKNOWLEDGMENTS

We are grateful to Jane Hillston and Stephen Gilmore for helpful discussions, comments and suggestions.

6. REFERENCES

[1] O. E. Akman, M. L. Guerriero, L. Loewe, and C. Troein. Complementary approaches to understanding the plant circadian clock. In *FBTC*, volume 19 of *EPTCS*, pages 1–19, 2010.

[2] L. Cardelli and P. Gardner. Processes in space. *Theor. Comput. Sci.*, 431:40–55, 2012.

[3] A. Compagnoni, M. Dezani-Ciancaglini, P. Giannini, K. Sauer, V. Sharma, and A. Troina. Parallel BioScape: A Stochastic and Parallel Language for Mobile and Spatial Interactions. In *MecBic'12, Newcastle, UK*, volume 100 of *EPTCS*, pages 101–106. Open Publishing Association, 2012.

[4] A. Compagnoni, V. Sharma, Y. Bao, P. Bidinger, L. Bioglio, E. Bonelli, M. Libera, and S. Sukhishvili. BioScape: A Modeling and Simulation Language for Bacteria-Materials Interactions. In *CS2Bio'12, Stockholm, Sweden*, volume 293 of *ENTCS*, pages 35–49, 2013.

[5] A. B. Compagnoni, P. Giannini, C. Kim, M. Milideo, and V. Sharma. A calculus of located entities. In *DCM'13, Buenos Aires, Argentina*, volume 144 of *EPTCS*, pages 41–56, 2014.

[6] J. Goubault-Larrecq, C. Palamidessi, and A. Troina. A probabilistic applied pi-calculus. In *APLAS*, volume 4807 of *LNCS*, pages 175–190. Springer, 2007.

[7] O. M. Herescu and C. Palamidessi. Probabilistic asynchronous pi-calculus. In *FoSSaCS*, volume 1784 of *LNCS*, pages 146–160. Springer, 2000.

[8] S. Jang, K. Oishi, R. Egbert, and E. Klavins. Specification and simulation of multicelled behaviors. *ACS Synthetic Biology*, 22(24):3067–3074, July 2012.

[9] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[10] C. Priami. Stochastic pi-Calculus. *Computer Journal*, 38(7):578–589, 1995.

[11] T. Rudge and J. Haseloff. A computational model of cellular morphogenesis in plants. In *ECAL 2005, Canterbury, UK*, volume 3630 of *Lecture Notes in Computer Science*, pages 78–87. Springer, 2005.

[12] D. Stewart, S. Gilmore, , and M. A. Cousin. FM-Sim : A hybrid protocol simulator of fluorescence microscopy neuroscience assays with integrated bayesian inference. In *Workshop on Hybrid Systems Biology*, 2014, to appear.

[13] D. Varacca and N. Yoshida. Probabilistic pi-calculus and event structures. *Electr. Notes Theor. Comput. Sci.*, 190(3):147–166, 2007.